INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 821

ON THE EXPRESSION OF GOVERNMENT AND BINDING PRINCIPLES BY CONTEXTUAL DISCONTINUOUS GRAMMARS

Patrick SAINT-DIZIER

AVRIL 1988

RR 821

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

PUBLICATION INTERNE N° 397

MARS 1988 - 14 PAGES

Patrick SAINT-DIZIER

ON THE EXPRESSION OF GOVERNMENT AND BINDING PRINCIPLES BY CONTEXTUAL DISCONTINUOUS GRAMMARS

Abstract :

In this short text, we explore the transfer of some principles of Government and Binding Theory to Natural Language Understanding. Several attractive formal and computational properties of Government and Binding theory are emphasized and illustrated. A logic-based grammar formalism, Contextual Discontinuous Grammars, is presented and its use to specify Government and Binding principles is shown.

DE L'EXPRESSION DE PRINCIPES DE LA THEORIE GOUVERNEMENT ET LIAGE PAR LES GRAMMAIRES CONTEXTUELLES DISCONTINUES

Résumé :

Nous explorons ici le transfert de certains résultats de la théorie linguistique de Gourvernement et Liage vers le traitement automatique du langage naturel. A cette fin, nous introduisons les grammaires contextuelles discontinues qui permettent de définir des relations et des contraintes entre constituants non-contigüs dans une structure.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

On the Expression of Government and Binding Principles by Contextual Discontinuous Grammars

Patrick Saint-Dizier

March 4, 1988

1 Government and Binding and Natural Language Understanding

In the traditional Transformational framework, natural language sentences are generated by means of two components: a base component that describes the basic form of sentences and a transformational component designed to build more complex structures from simpler ones. Due to this latter component, we can, theoretically, achieve a clear separation between syntax and semantics because transformations often permit the generation of several sentences with an equivalent meaning.

Transformations in the standard theory are very complex and have to deal with very diverse phenomena like: construction of passive forms, NP topicalization and extraposition, Wh-movement, cross-over effects, construction of cleft-sentences, subjacency constraints, etc... The result is a proliferation of rules and a large set of constraints to prevent interferences between

Current address: IRISA, Campus de Beaulieu, 35042 RENNES - Cedex FRANCE

rules. It is also extremely difficult to incorporate semantic rule constructions into these rules.

Government and Binding theory (Chomky 82, 86), noted hereafter as GB, is a complete revision of this baroque set of rules and constraints, acheiving a much greater expressive power and explanatory adequacy. GB theory is composed of a very small base component, a single movement rule and a small set of principles whose role is to control the power of the movement rule. We won't argue extensively here in favor of GB, but it seems that the transfer of some principles and results of GB to Natural Language Understanding is worth investigating. GB theory exhibits a greater clarity, expressivity, ease of understanding and extensiveness (in spite of several points which are still somewhat obscure). Futhermore, the current formalization of GB has several attractive computational properties such as:

- concision and economy of means,
- high degree of parametrization,
- modularity (independence of filtering principles),
- declarativity (no order in the application of rules),
- absence of intermediate structures (e.g. deep structure).

In GB, grammaticality of a sentence is based upon the existence of a well-formed annotated surface form of that sentence. Thus, no real movements occur and some difficult computational problems (like circularity) are avoided.

In this short document, we show how some principles of GB theory can be fruitfully transfered to Natural Language Understanding. We introduce X-bar syntax, theory of movement, bounding theory, Θ-theory and case filtering. They are given a logical characterization and a specific logic-based grammar formalism has been developed for that purpose and is presented here. We call it Contextual Discontinuous Grammars (CDGs hereafter). It is at the origin of DISLOG, an extension to PROLOG which permits to deal with long-distance relations in a structure in a declarative and transparent way. In (Saint-Dizier 87), we show that this language is well adapted to deal with AI applications like planning, scheduling and problems involving traversal constraints.

Our general goal is not to build a complete implementation of GB theory but to borrow from it some clear and simple principles and formulations

in order to build more general and more adequate natural language understanding systems. For that purpose, we also have to include elements from other Computational Linguistics and Artificial Intelligence approaches to deal, for example, with the lexicon (which should be very informative), lexical semantics, feature systems, logical form construction and interactions with pragmatic knowledge. We also introduce methodological elements.

Up to now, very few and always partial works have been carried out on GB theory. Let us mention (Stabler 88), (Berwick 86) and (Brown et al. 88). There is, however, an increasing interest for this apraoch.

In the following sections, we introduce X-bar syntax and its implementation, movement theory, Θ-theory and CDGs and, finally, bounding theory. We have investigated different parsing strategies that we briefly mention.

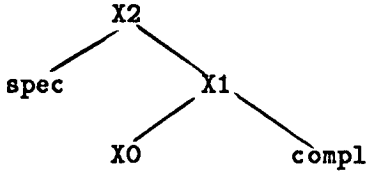
2 The Base component

The Base component is based on current X-bar syntax theory. It permits to build basic sentences. The rules are very concise and highly parametrized. Here are the rules of the Base component:

$X^2 \rightarrow spec, X^1$.
$X^1 \rightarrow X^0, compl$.
$X^1 \rightarrow compl, X^0$.
$X^1 \rightarrow adjunct, X^1$.
$X^1 \rightarrow X^1, adjunct$.

In these rules, X stands for the clausal categories COMP or INFL (Complementizer and Inflectional) or for one of the lexical categories V (verb), N (noun), A (adjective) or P (preposition). The index i in X^i indicates the bar level. The highest level is 2. It corresponds to the more familiar categories VP, NP, etc... X is the head of the rule and controls the syntactic and semantic feature agreement. There is a direct inheritance link between the two X categories in a rule.

Consequently, the global structure of each category is:



Each spec and compl may be filled in with another X^2 but each category has its own spec and compl. For example, the spec of an N is a determiner and its compl is a relative clause. The spec of a V is an auxiliary and its compl is a direct complement. Some terminal element X^0 may also have an empty realization.

The adjunction of complements to a verb, of adjectives to a noun is not well-defined in GB. In our framework it is the role of the lexicon to explicit these relations by means of features, which are then controlled and percolated by means of a logical feature system (Saint-Dizier 86). Our lexicon is a set of logical axioms expressed in Prolog. Using a certain lexical entry (i.e. a word and the specification of its syntactic and semantic environment in a sentence) is proving that it is appropriately used.

In a simplified way, here is the PROLOG implementation of a base rule. The general form is that of a Definite Clause. Each non-terminal has four arguments:

- Cat: syntactic category
- Bar: Bar level
- F: feature structure
- LF: associated logical form.

A symbol is noted as:
xp(Cat, Bar, F, LF).
The first rule above is written in PROLOG as:

```
xp(Cat,2,F,LF) -->
   {type_of_spec(Cat,Cat1,Bar1)},
   xp(Cat1,Bar1,F1,LF1),
   xp(Cat,1,F,LF2),
   {agreement_control(F1,F,Cat,Cat1),
   logic(LF1,LF2,Cat,Cat1,LF) }.
```

type_of_spec refers to a small set of data which gives the basic specifier and its bar level for each xp category:

```
type_of_spec(n,det,0).
type_of_spec(v,aux,0).
```

logic refers to semantic representation construction rules. It is based on the logical representation of each subconstituent and their syntactic category. *logic* also checks semantic feature agreement. We thus have a clear separation between the structural description and the semantic processing.

We produce a simple, PROLOG-like representation including quantification (Saint-Dizier 86) that we will not examine here. At this level, our goal is to incorporate the construction of structures of Situation Theory (Barwise and Perry 83). We would only like to mention at that level that the very regular rule format of the form *modifier-modifiand* or the reverse and the different levels of modification introduced by *spec*, *compl* and *adjunct* permit the definition of simple, standard and parametrized semantic rules.

For example, for a determiner and a noun, if F1 is the representation of the determiner and if F2 is the representation of the noun and X the variable introduced by the determiner, then the semantic composition rule for our logical representation is:

```
logic(F1,f(X,F2),det,n,f(X,quant([F1,X],F2,F3))).
```

Notice that the notation f(X,F2) is a mean to link X and F2. This is a direct translation of a λ-expression in PROLOG. F3 is the representation of the remainder of the sentence. In some cases, features and contraints are introduced in semantic composition rules when several representations are possible.

Finally, syntactic categories xp with bar level 0 are dealt with by lexical insertion rules like (in a somewhat simplified way):

```
xp(n,0,F,LF) --> noun(F,LF).
```

The base component is thus very concise and modular. It comprises 6 rules (left-recursion elimination apart), a database of specifiers, complements and adjunct for each syntactic category, semantic representation construction rules (about 20 rules) and lexical insertion rules. This Base component is associated with a lexicon containing detailed subcategorization data (prepositions and type of complements, ...). In spite of the highly parametrized level of rules, the bottom-up implementation we have carried out of this grammar turns out to be efficient (cf. section 2.3).

3 Movement rules

3.1 Movement theory

GB postulates a single movement rule, move- α , controlled by principles and filters. This very general rule states:

Move any constituent α to any position

The most immediate contraints on that rule specify that:

- (a) α has to move to an empty position (Empty Category Principle) of the same category than α,
- (b) α is moved to the left by adjunction or substitution,
- (c) an N without case must move to a position where it is assigned case (Case Filter),
- (d) each N has one and only one Θ -role (Thematic role: agent, goal, instrument,...).

Move- α and these constraints are clearly too abstract and computationally untractable to be directly implemented. We now introduce Contextual Discontinuous Grammars (CDGs for short) that we have specifically designed to express movement rules and more generally to express relations between non-contiguous elements in a structure.

3.2 Contextual Discontinuous Grammars

CDGs are essentially motivated by two factors:

- (1) move- α is too abstract and needs to be specified for each class of movement, permiting in the same time to directly encode constraints (a) and (b) above. No generality or explanatory power is lost by this specification which we consider as a simple instanciation.
- (2) Since no real movements are postulated (recall that grammaticality of a sentence is based upon a well-formed annotated surface form of that sentence) it is no longer necessary to have movements of strings as we did in Gapping Grammars (Dahl and Saint-Dizier 86).

What we only need is to establish relations between non-contiguous parts of this annotated surface form in order to control that constraints on movements are satisfied and also in order to percolate features. We now turn

to informally introduce CDGs. More formal details can be found in (Saint-Dizier 87, 88) about procedural and declarative semantics of CDGs and DISLOG, their logic programming counterpart.

A contextual discontinuous rule has the form of a set (or an unordered tuple) of context-free rules:

```
{ a --> a1,  b --> b1,  ...  , n --> n1 }
```

which can be paraphrased by:

Somewhere in the parsing tree a is rewritten into a1, b into b1, ..., and n into n1, with the same substitutions used for all identical variables occuring in the context-free rules of that CDG rule

In this rule, a, b, ..., n are non-terminal symbols and a1, b1, ..., n1 are a sequence of non-terminal and terminal symbols. Symbols are usually augmented by arguments. A set with a single rule :

```
{ a --> a1 }.
```

is equivalent to a context-free rule.

In the above formulation, derivations are free to appear in any order. However, it seems that at least some partial ordering is quite often required when one wants to describe the grammar of a language. The above notation can then be associated with Linear Precedence restrictions. For instance, if in the following tuple:

```
{ a --> a1,  b --> b1,  c -> c1  , d --> d1 }
```

we want to impose the linear restriction:

$a \rightarrow a1$ has always to be to the left of or above $c \rightarrow c1$ and $d \rightarrow d1$ in the parse tree,

then we add the restriction:

```
a < c , a < d .
```

Thus, the general form of a contextual discontinuous rule is:

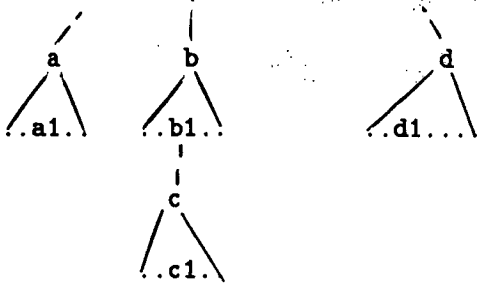
```
{ a --> a1,  b --> b1, ... , n --> n1 } x < y, .... ,  z < t.
```

where x, y, z and t are non-terminal symbols which appear in the left-hand part of the rules given in the tuple. We make the assumption that there is no ambiguity in the identification of symbols subject to a linear precedence restriction, otherwise rules can, for example, be indexed.

There are, in fact, a number of cases where the order of context-free rules in a tuple is complete. In that case, we adopt the notation:

```
a --> a1 / b --> b1 / c --> c1 / d --> d1.
```

which means that, in a parsing tree where this rule is used, the subtree representing the expansion of $a \rightarrow a1$ will have to be to the left of or above the subtrees representing the expansion of the rules $b \rightarrow b1, c \rightarrow c1$ and $d \rightarrow d1$, and so forth for b and c. An acceptable configuration would be, for instance:



Another aspect of CDGs is the adjunction of modalities. The main modality we need is the possibility, noted as m . This modality can be attached to any rule a_i in a CDG. It allows this rule to be used any number of time, including zero:

$\{r_1, ..., m(r_i), ...\}$

Introducing modality m considerably enhance, for example, the ease of writing compilers (Saint-Dizier 87) and the treatment of parasitic gaps and coordination.

3.3 Movement theory and CDGs

We have identified so far 8 CDG rules. We now present the relativization of a subject or a direct object in French. The corresponding annotated surface form is:

$[_{COMP}\ PRO_i$.....$[_{N2}\ trace_i\ ]\ ...\ ]$

as in:

$[_{COMP}\ THAT_i\ John\ met\ trace_i\ yesterday\ ]$

In relative clause construction, a N is pronominalized and moved to the left and adjoined to a COMP node. A trace of the N is left behind and co-indexed with the pronoun PRO.

We then have the following CDG rule (feature representation is simplified for the sake of clarity):

```
{ xp(comp,0,pro,_) --> xp(n,0,pro,I),
                       xp(comp,0,_,I). ,
xp(n,2,F,I) --> []. }
```

Notice that using the same variable in different rules of a CDG rule permits to transfer feature values in a declarative and simple way.

Here is another CDG rule which deals with passivation of affirmative sentences. In this case there are two phenomena: the direct object is "moved" to the front of the sentence and the subject becomes an object introduced in French by the preposition "par". The direct object is substituted to the empty string resulting from the derivation of the $COMP^0$ and the subject is right-adjoined to the verb. Syntactic features of the verb are also updated. Here is the passivation CDG rule, where features have been simplified for the sake of clarity:

```
{ xp(v,0,T2,F2) --> xp(v,0,T2,F2), [par], xp(n,2,T,F). ,
  xp(comp,0,T,F) --> xp(n,2,T1,F1).  ,
  xp(n,2,T1,F1) --> [].  ,
  xp(n,2,T,F) --> [].  }
```

Notice that no element of semantic representation is elaborated at that level since it is only a movement rule. However, for instance for passive constructions, we keep track of the passive voice use since it can be meaningful in some situations. The same remark holds when we use CDGs out of GB theory to deal, for example, with sentence temporal and conditional connectors.

The problem of associating a (partial) semantic representation to CDG rules is not a trivial matter in general. It is quite simple within GB theory of movements, but when we extend the linguistic coverage to constructions not (yet?) accounted for in GB then we often have to take into account other factors in the sentence. To illustrate this problem of context-sensitivity, consider, for example, temporal expressions. The meaning of sentences connected by temporal expressions does not depend only on these expressions but also on the tense and the voice of verbs in that sentence. Thus, all those non-contiguous expressions and features have to intervene and to be expressed in a single CDG rule to determine an appropriate semantic representation.

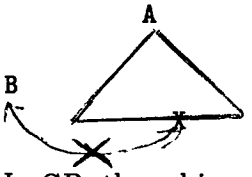
3.4 A PROLOG implementation of CDGs

We have implemented an extension to PROLOG, called DISLOG (programming in logic with discontinuities), which interprets CDG rules in PROLOG. Once a rule in a CDG is used then the other rules are stored in a list with appropriate substitutions and percolated throughout the parse. A sentence is well-formed iff all the stored rules have been used.

We have first carried out a top-down implementation (Saint-Dizier 87). This implementation is however not very efficient. More recently, we have implemented a bottom-up strategy using the left-corner technique which turns out to be of a good efficiency (a 15 word sentence is parsed in about 4 seconds CPU time on a Sun workstation). Heuristics are under study to improve this technique, using preferences among rules. A Chart-parsing strategy is also under study.

4 Bounding Theory

This theory states constraints on the way to move constituents in a structure or, in other terms, it limits the freedom of establishing relations between non-contiguous elements of a structure. It is expressed in terms of domains over the boundaries of which constituents cannot be moved in one time (i.e. intermediate landing sites are needed). Roughly speaking, if A is a bounding node, then the domain of A is the subtree it is the root of. Then, no constituent X in the domain of A can establish relations with any element B outside the domain of A:



In GB, the subjacency constraint is a constraint of that type. We have somewhat simplified it in order to make it computationally tractable. This simplification is justified by recent works which tend to extend the power of this constraint. In French bounding nodes are N^2 and S^1.

The DISLOG interpretor treats special facts like:

```
bounding_node(N2).
```

```
bounding_node(S1).
```

as a special directive. All CDG rules which have begun to be used in the domain of a bounding node have to be fully used within that domain. This constraint is straightforward to implement in Prolog by means of lists.

Bounding theory turns out to be also of much interest for semantic representation computation. For example, quantifier raising in logical representations is bounded by bounding nodes like N^2 , $INFL^1$, conjunction, modal and temporal connector nodes.

5 Epilogue

In this short document we have shown how principles of Government and Binding theory can be transfered to Natural Language Understanding. We have defined Contextual Discontinuous Grammars for that purpose and proposed an extension to Prolog called Dislog, programming in logic with discontinuities. Semantic composition rules have also been introduced.

Finally, we have emphasized the role of bounding theory for syntactic parsing as well as for semantic compositional rules.

We are now integrating a large scale lexicon to the parser. The form of this lexicon as well as the modes of access are a complex problem especially as soon as one wants to include semantic and subcategorization information.

R E F E R E N C E S

Barwise, J. and Perry, J., *Situations and Attitudes*, MIT Press, 1983.

Berwick, R., Weinberg, A., *The Grammatical Basis of Linguistic Performance*, MIT Press, 1986.

Brown, C., Pattabhiraman, T., Massicotte, P., Towards a Theory of Natural Language Generation, in: *Natural Language Understanding and Logic Programming II*, V. Dahl and P. St-Dizier Edts, North Holland, 1988.

Chomsky, N., *Lectures on Government and Binding*, Foris Pub., Holland, 1982.

Chomsky, N., *Barriers*, Linguistic Inquiry Monograph Nb. 13, MIT Press, 1986.

Dahl, V. and Saint-Dizier, P., Constrained Discontinuous Gramars - A linguistically motivated tool for natural language, to appear in: *New Computing Generation Journal*, 1988.

Saint-Dizier, P., Expression of Features in Logic-Based Grammars, *Computational Intelligence*, Vol 2-1, 1986 (a).

Saint-Dizier, P., An Approach to Natural Language Semantics in Logic Programming, in: *Journal of Logic Programming*, vol 3-4, 1986 (b).

Saint-Dizier, P., DISLOG: Programming in Logic with Discontinuities, Simon Fraser University research report, lccr-tr 87-13, Vancouver, BC, Canada. 1987.

Saint-Dizier, P., Contextual Discontinuous Grammars, in: *Natural Language Understanding and Logic Programming II*, V. Dahl and P. Saint-Dizier Edts, North Holland Pub., 1988.

Stabler, E., Parsing with explicit Representations of Syntactic Constraints, same reference as above, 1988.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique